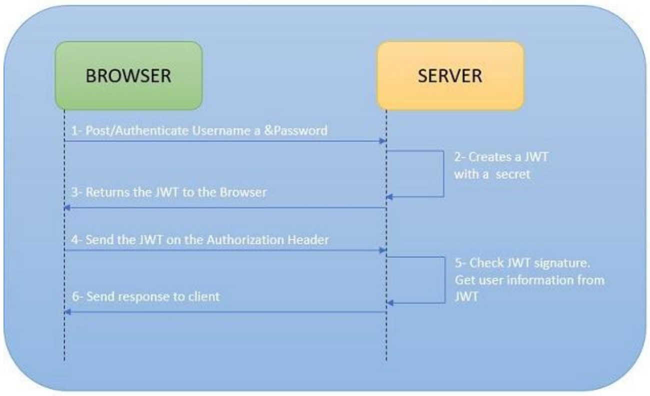
Q: What is JWT? How to implement?

JSON Web Token (JWT) is an open standard (RFC 7519) that specifies a compact and self-contained way of transmitting information securely as a JSON object between parties. This information can be verified and trusted as it has been digitally signed. It can also hold all the user's claim, like authorization information, so that the service provider does not need to access the database to validate user roles and permissions for each request; data is extracted from the token.

Q: What is Workflow of JWT?


BROWSER
SERVER
1- Post/Authenticate Username a &Password
2- Creates a JWT with a secret
3- Returns the JWT to the Browser
4- Send the JWT on the Authorization Header
5- Check JWT signature. Get user information from JWT
6- Send response to client

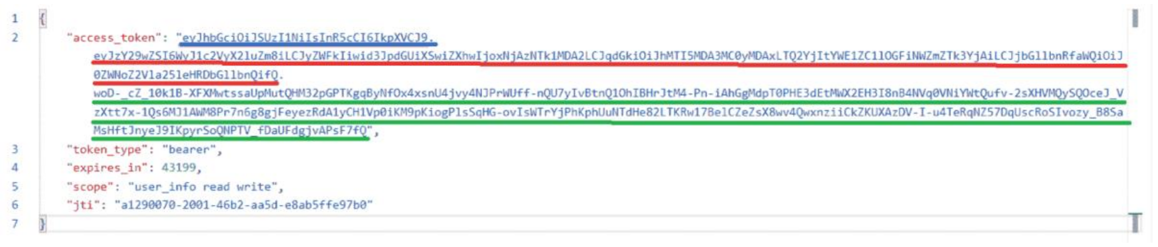
- *Customers sign in by submitting their credentials to the provider.*
- *Upon successful authentication, it generates JWT containing user details and privileges for accessing the services and sets the JWT expiry date in payload.*
- *The server signs and encrypts the JWT if necessary and sends it to the client as a response with credentials to the initial request.*
- *Based on the expiration set by the server, the customer/client stores the JWT for a restricted or infinite amount of time.*
- *The client sends this JWT token in the header for all subsequent requests.*
- *The client authenticates the user with this token. So we don't need the client to send the user name and password to the server during each authentication process, but only once the server sends the client a JWT.*

Q: What is the structure of JWT?

JWT consists of 3 parts - Header.Payload.Signature

Header
Payload
Signature

It generate JWT token as in the form of a.b.c which represent header.payload.signature



```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
      eyJzY29wZSI6WyJ1c2VyX2luZm8iLCJyZWFkIiwid3JpdGUiXSwiZXhwIjoxNjAzNTk1MDA2LCJqdGkiOiJhMTI5MDA3MC0yMDAxLTQ2YjItYWE1ZC11OGFiNWZmZTk3YjAiLCJjbGllbnRfaWQiOiJ
      0ZWNoZ2Vla251leHRDbG1lbnQifQ.
      woD-_cZ_10k1B-XFXMwtssaUpMutQHM32pGPTKgqByNfOx4xsnU4jvy4NJPrWUff-nQU7yIvBtnQ1OhIBHrJtM4-Pn-iAhGgMdpT0PHE3dEtMWX2EH3I8nB4NVq0VNiYWtQufv-2sXHVMQySQOceJ_V
      zXtt7x-1Qs6MJ1AWM8Pr7n6g8gjFeyezRdA1yCH1Vp0iKM9pKiogPlsSqHG-ovIsWTrYjPhKphUuNTdHe82LTKRw17BelCZeZsX8wv4QwxnziiCkZKUXAzDV-I-u4TeRqNZ57DqUscRoSIvozy_B8Sa
      MsHftJnyeJ9IKpyrSoQNPTV_fDaUFdgjvAPsF7fQ",
    "token_type": "bearer",
    "expires_in": 43199,
    "scope": "user_info read write",
    "jti": "a1290070-2001-46b2-aa5d-e8ab5ffe97b0"
}
```

Q: What is expiration date of JWT?

The JWT access token is only valid for a limited period of time. Using an expired JWT would cause the operation to fail. This value is normally 1200 seconds or 20 minutes.

Q: How do we specify expiration date of JWT?

```
private String doGenerateToken(Map<String, Object> claims, String subject) {
                    return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))
                                        .setExpiration(new Date(System.currentTimeMillis() + jwtExpirationInMs)).signWith(SignatureAlgorithm.HS512, secret).compact();
          }
```

Q: What are the advantages of JWT?

1. ***Good Performance:*** *JWT itself contains all information, so we don't have to go to Authorization server to get the user's information to verify whether user is valid or not.*
2. ***Portable:*** *Allow to use multiple backends with single access token.*
3. *It is Very Mobile Friendly, because cookies are not required.*
4. *JWT contains expiration date as a claim that can be used to determine when the access token is going to expire.*
5. *It's very secure way to validate the user information, as it's digitally signed.*

6. *It's digitally signed, so if anyone updates it the server will know about it.*
7. *It is most suitable for Microservices Architecture.*
8. *It has other advantages like specifying the expiration time.*

Q: What is OAuth and JWT?

JWT is essentially a token format. JWT is a token that can be used as part of the OAuth authorization protocol. Server-side and client-side storage are used in OAuth. If you want to make a proper logout, you'll need to use OAuth2. Authentication with a JWT token does not allow you to logout.

Q: Is JWT required while communicating over HTTPS?

The JWT is signed with public/private key pairs to ensure that the sender is authenticated and that the payload has not been tampered with. The JSON Web Token, on the other hand, is in plain text.

*To **encrypt** communication, we will require **SSL/HTTPS**. Attackers can intercept network traffic without SSL/HTTPS and extract the JWT, making your application vulnerable to man in the middle attacks.*

Q: Why JWT is a stateless authentication?

JSON Web Tokens (JWT) are called stateless because the authorizing server doesn't need to keep track of anything, the token is all that's required to verify a token bearer's authorization. In stateless authentication, no need to store user information in the session.

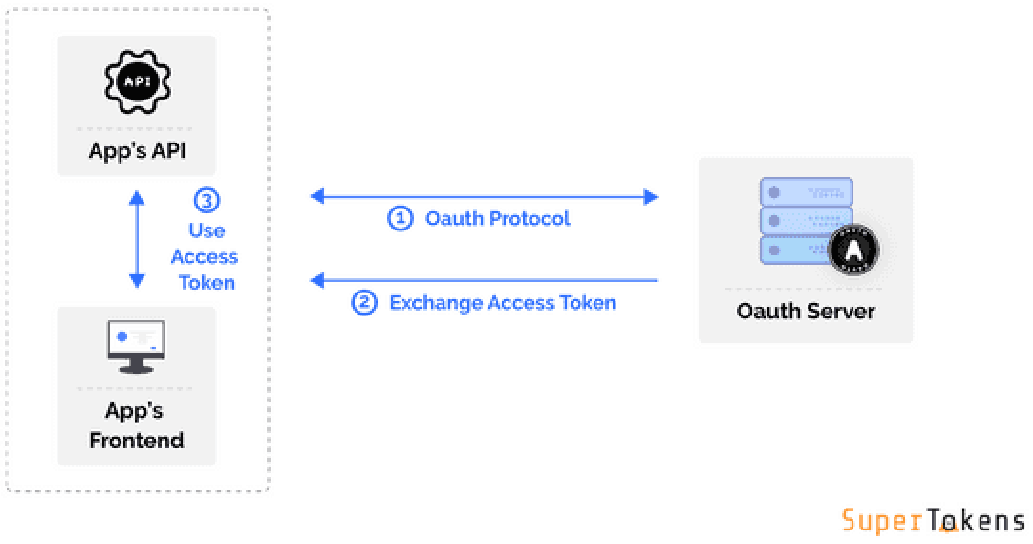
Q: Does JWT token contain password?

The JWT comprises encoded user information as well as a signature that is checked when decoded to confirm that the token has not been tampered with. After the JWT has been confirmed, instead of sending the user their forgotten password, your application can safely allow them to generate a new password.

Q) What is OAuth?

OAuth, short for "Open Authorization," is an open standard protocol that allows secure authorization in a simple and standardized way for web, mobile, and desktop applications. It's like a valet key for your car – it gives limited access to your resources without handing over the full set of keys.

How does OAuth Work?



OAuth works by delegating user authentication to the service that hosts the user account and authorizing third-party applications to access that user account. Here's a simplified flow:

1. *The user initiates a request to the application.*
2. *The application redirects the user to the OAuth provider.*
3. *The user authenticates with the OAuth provider.*
4. *The OAuth provider asks the user to authorize the application.*
5. *The user grants permission.*
6. *The OAuth provider sends an authorization code to the application.*
7. *The application exchanges the code for an access token.*
8. *The application uses the access token to access protected resources.*

OAuth supports several authorization flows, including the Authorization Code Flow, Implicit Flow, and Client Credentials Flow. Each has its use cases, but the Authorization Code Flow is generally recommended for most scenarios due to its security benefits.

Q) Key Differences Between OAuth and JWT

Difference 1 - Use Cases and Purpose

- ***OAuth**: Primarily used for authorization and delegating access. It's great when you need to grant third-party applications limited access to user resources without sharing credentials.*
- ***JWT**: Used for secure information exchange and authentication. It's perfect for stateless authentication in web applications, especially in microservices architectures.*

Difference 2 - Implementation and Complexity

- ***OAuth**: Involves a multi-step process with different roles (client, resource owner, authorization server, resource server). It's more complex but offers more flexibility.*
- ***JWT**: Simpler to implement as it's self-contained. The token itself carries all necessary information, making it easier to use in stateless applications.*

Difference 3 - Security and Management

- ***OAuth**: Offers fine-grained access control through scopes. Tokens can be easily revoked, enhancing security.*
- ***JWT**: Relies on cryptographic signatures for security. Once issued, JWTs are valid until they expire, which can be a security concern if not managed properly.*